REMARKS

I. Overview

These remarks are set forth in response to the Non-Final Office Action. Presently, claims 1, 3 through 13 and 41 are pending in the Patent Application. Claims 1 and 41 are independent in nature. In the Non-Final Office Action, the Examiner has objected to the form of claims 1 and 41 and has suggested claim amendments to overcome those objections. In response, Applicants have applied Examiner's helpful suggestions. Additionally, claims 1, 3, 4 and 6 through 13 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,442,754 to Curtis et al. (Curtis) in view of U.S. Patent No. 6,725,452 to Te'eni and further in view of U.S. Patent No. 6,675,382 to Foster et al. (Foster). Further, claim 5 has been rejected under 35 U.S.C. § 103(a) as being unpatentable over Curtis, Te'eni and Foster, and further in view of U.S. Patent No. 6,918,112 to Bourke-Dunphy et al. ("Bourke-Dunphy"). Finally, claim 41 has been rejected under 35 U.S.C. § 103(a) as being unpatentable over Curtis in view of Foster.

II. The Applicants' Invention

The Applicants' invention relates to deploying software components, for instance into an enterprise environment. In an embodiment of Applicants' invention, components that had previously been installed and components that are to be installed are identified. Conflicts between such components are then identified. A user may be notified and provided with options in the event of an identified conflict. One option presented to the user include aborting the installation. Another option is to continue the installation. In the latter circumstance, an entry may be made in a log indicative of the conflict and of the continuation of the installation.

III. Rejections Under 35 U.S.C. § 103(a)

Examiner has rejected claims 1, 3, 4 and 6 through 13 as being merely an obvious variation over Curtis, Te'eni and Foster. Further, examiner has rejected claim 41 also as being merely an obvious variation over only Curtis and Foster. Claim 5 has been rejected based upon the combination of Curtis, Te'eni, Foster and Bourke-Denphy. Of note, the "Examination Guidelines for Determining Obviousness Under 35 U.S.C. 103 in View of the Supreme Court Decision in KSR International Co. v. Teleflex Inc.," 73 Fed. Reg. 57,526 (2007) (hereinafter the Examination Guidelines) specify the burden under which the Examiner is to conduct an analysis under 35 U.S.C. § 103(a). Section III is entitled "Rationales To Support Rejections Under 35 U.S.C. 103." Within this section is the following quote from the Supreme Court: "rejections on obviousness grounds cannot be sustained by merely conclusory statements; instead there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness." KSR Int'l Co.,127 S. Ct. 1727, 1741 (2007) (quoting In re Kahn, 441 F.3d 977, 988 (Fed. Cir. 2006)). Examiner appears to have articulated a reasoning supporting the rejection of claims 1, 3 through 13 and 41 in the Non-Final Office Action.

Referring to the first column on page 57,529 of the Examination Guidelines, the following is a list of rationales that may be used to support a finding of obviousness under 35 U.S.C. § 103:

> (A) Combining prior art elements according to known methods to yield predictable results;
> (B) Simple substitution of one known element for another to obtain predictable results;
> (C) Use of known technique to improve similar devices (methods, or products) in the same way;
> (D) Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;
> (E) "Obvious to try" - choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success;
> (F) Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations would have been predictable to one of ordinary skill in the art;
> (G) Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

Upon reviewing the Examiner's analysis in the paragraph spanning pages 3 through 17 of the Non-Final Office Action, the Examiner appears to be employing rationale (A). If the Examiner is not relying upon rationale (A), Appellants request that the Examiner clearly identify the rationale, as described in the Examination Guidelines, being employed by the Examiner in rejecting the claims under 35 U.S.C. § 103.

The Examination Guidelines with respect to rationale (A) set forth a precise process for which the Examiner must follow in order to establish a prima facie case of obviousness under 35 U.S.C. § 103(a). Specifically, to reject a claim based on this rationale, Office personnel must resolve the Graham factual inquiries. Thereafter, Office personnel must then articulate the following:

> (1) **a finding that the prior art included each element claimed**, although not necessarily in a single prior art reference, with the only difference between the claimed

invention and the prior art being the lack of actual combination of the elements in a single prior art reference;
(2) a finding that one of ordinary skill in the art could have combined the elements as claimed by known methods, and that in combination, each element merely would have performed the same function as it did separately;
(3) a finding that one of ordinary skill in the art would have recognized that the results of the combination were predictable; and
(4) whatever additional findings based on the Graham factual inquiries may be necessary, in view of the facts of the case under consideration, to explain a conclusion of obviousness.

With respect, Examiner has not adequately articulated a finding that the prior art included each element claimed with the only difference between the claimed invention and the prior art being the lack of actual combination of the elements in a single prior art reference.

In this regard, the *lengthy* claim 1 recites a computer-implemented method for a pre-deployment analysis of software components of an application prior to deployment of the application. For the convenience of Examiner, claim 1 as amended is reproduced herein as follows:

1. A computer implemented method for a pre-deployment analysis of a plurality of software components of an application prior to deployment of the application, comprising:

including, in an installation package for the application, **a data structure that provides, <u>for each</u> of the plurality of software components from the application**, a software component deployment dependency data, an indication of necessary software components for an operation of each of the plurality of software components being installed, **and an indication of incompatibility with a previously installed software component**; and

loading the installation package into a memory connected to a computer; and,

using the computer connected to the storage and a program installed in a memory of the computer so configured by the installation package, performing the steps of:

determining a first plurality of software components previously installed on a system;

determining a second plurality of software components to be installed on the system;

accessing a third plurality of software component deployment dependency data;

determining a fourth plurality of software components suitable for parallel installation;

determining an order in which the fourth plurality of software components can be grouped for a fifth plurality of parallel installations;

accessing a sixth plurality of metadata from the data structure regarding the second plurality of software components to be installed and accessing a seventh plurality of metadata regarding the first plurality of software components previously installed; and

analyzing the sixth plurality of metadata to determine an eighth plurality of potential conflicts between the second plurality of software components to be installed and the first plurality of software components previously installed on the system;

wherein the pre-deployment analysis allows the second plurality of software components to be installed in parallel and in a sequence of groups; and

wherein an installation time for the application is reduced.

Integral to claim 1 is the inclusion within the installation package for each software component of the application, of an indication of incompatibility with a <u>previously installed</u> software component.

On pages 5 and 6 of the Non-Final Office Action, Examiner concedes that Curtis does not disclose the indication of incompatibility with a previously installed software component. However, Examiner refers to Te'eni, column 1, lines 61 through 64 and column 5, lines 10 through 17 for this teaching. Specifically, Examiner stated,

Te'eni discloses:

an indication of incompatibility with a previously installed software component *(see Column* 1: 61-64, *"When performing the predefined procedures necessary for an upgrade to be implemented frequently dependency conflicts may arise among the components present and the components to be installed."; Column* 5: *10-17, "Virtual upgrade module* 36 *creates upgrade processes for the following tasks that are performed sequentially: (a) to collect all the information necessary for the dependency analysis and the dependency conflicts resolving process such as the relevant component data information units from component data table 28, the encoded dependency rules from xor-rules table* 32 *and from add-remove rules table 34 module ...* ")

In referring to column 1, lines 61 through 64 of Te'eni, it is clear that only a general statement exists that conflicts arise during installation of components. Column 5, lines 10 through 17 only add that an upgrade process is created for a sequence of tasks that includes collecting information for a "dependency conflicts resolving process". For the convenience of Examiner, column 5, lines 10 through 35 are reproduced verbatim as follows in order to provide enhanced context for the small portion cited by Examiner:

> The set of application programs or application modules includes virtual upgrade module 36, conflict resolver module 40 and component installer module 42. Virtual upgrade module 36 is the manager of the virtual upgrade process activated by the user via user interface (GUI) 20. Virtual upgrade module 36 creates upgrade processes for the following tasks that are performed sequentially: (a) **to collect all the information necessary for the dependency analysis and the <u>dependency conflicts resolving process</u>** such as the relevant component data information units from component data table 28, the encoded dependency rules from xor-rules table 32 and from add-remove rules table 34 module, (b) **<u>to activate conflict resolver module 40</u> in order <u>to check for potential dependency conflicts</u> and to resolve any dependency conflicts that might arise as a result of the planned installation process**, (c) to initiate the downloading of the required component objects from the supporting server system or from any other source, and (d) to perform a genuine installation of the selected component objects from component objects table 44.

Thus, it is a "conflict resolver module 40" in Te'eni that checks for conflicts which is a natural result since there is not static, pre-stored information regarding known conflicts that exist between a software component to be installed and a software component already installed in an installation package for the components of the application. So much, however, is expressly required by the claim language of claim 1. Accordingly, the combination of Curtis, Te'eni and Foster do note provide a teaching to each element claimed in claim 1 as required by the Examination Guidelines, rationale (A).

As to even lengthier claim 41, a computer implemented method is provided for the use of a semantic model to increase the efficiency of deployment of an application to a target by maximizing parallel installation of application software components. Claim 41 as amended recites:

41. A computer implemented method of using a semantic model to increase the efficiency of deployment of an application to a target by maximizing parallel installation of application software components, the computer implemented method comprising:

accessing the semantic model to obtain a dependency information about software components of an application;

including a semantic model in an installation package of the application;

responsive to loading the installation package into a memory connected to a computer, using the computer so configured by the installation package to perform steps comprising:

storing a first record of each of a plurality of the software components that is to be deployed in a read file;

storing a second record of each of a plurality of previously installed software components in a registry file;

when the read file is available to deploy, examining the registry file and accessing the semantic model to obtain a plurality of dependency information indicating a plurality of relationships among the plurality of the software components to be installed in the target and among a plurality of previously installed software components;

using the dependency information to *group* the plurality of the software components into sets of software components *with like dependency levels*, wherein a first set of software components from amongst the sets of software components has no dependencies, a second set of software components from amongst the sets of software components has dependencies only on the first set of software components, and a third set of software components from amongst the sets of software components has dependencies only on the first and second sets of software components;

installing the first set of software components in parallel;

responsive to completing installation of the first set of software components, installing the second set of software components in parallel;

responsive to completing installation of the second set of software components, installing the third set of software components in parallel;

when a component is installed, updating the registry file;

when a conflict is identified, taking an appropriate action; and

displaying a progress report by labeling the plurality of the software components in the semantic model in a selected level of granularity.

On page 15 of the Non-Final Office Action, Examiner refers to column 13, lines 7 through 27 and 33 through 37 of Curtis for the important teaching of using dependency information in order to group software components into sets according to <u>like</u> dependency levels.

However, column 13 of Curtis only provides for the hierarchical arrangement of components and dependency data in a registry as provided by column 13 of Curtis reproduced herein as follows:

> FIG. 5 illustrates a data structure, which in preferred embodiments is maintained in the registry object or registry database 220, indicating installed programs and dependent components on which installed programs depend. In the embodiment of FIG. 5, the data structure is a <u>hierarchical arrangement of programs, file sets, and dependent components in the form of a directory tree.</u> A root directory includes a sub-directory for each installed program, indicating the program name and version. Each installed program includes a next level subdirectory for each file set of the installed program, indicating the file set name and version. **Each installed file set component has a Dependency subdirectory which includes information on each dependent component on which the file set and program depend in order to operate.** The dependency subdirectory would list the program name, version, fileset name, and fileset version for each program on which the fileset including the dependency subdirectory depends. The registry route 436 in the dependency object 400 indicates the location of the dependency directory in the registry file where the dependency information for a particular dependency object is maintained.

As will be apparent from a review of the recited portion of Curtis, no teaching can be found pertaining to the grouping of components into sets <u>with like dependency levels</u>. In that claim 41 requires this precise teaching, Examiner has not satisfied the requirements of rationale (A) set forth by the Examination Guidelines.

<u>V. Conclusion</u>

In that Curtis, Foster, Te'eni and Bourke-Denphy lack as a combination teachings claimed within independent claims 1 and 41, Applicants respectfully request the withdrawal of the rejections under 35 U.S.C. § 103(a) owing to the amended claims and the foregoing remarks. The Applicants request that the Examiner call the undersigned if clarification is needed on any matter within this Amendment, or if the Examiner believes a telephone interview would expedite the prosecution of the subject application to completion.

Respectfully submitted,

Date: July 6, 2009

/Steven M. Greenberg/

Steven M. Greenberg, Reg. No.: 44,725
Attorney for Applicant(s)
Carey, Rodriguez, Greenberg & Paul, LLP
950 Peninsula Corporate Circle, Suite 3020
Boca Raton, Florida 33487
Customer No. 46320
Tel: (561) 922-3845
Fax: (561) 244-1062